

Empirical Performance of the Approximation of the Least Hypervolume Contributor

Krzysztof Nowak¹, Marcus Märtens², and Dario Izzo¹

¹ European Space Agency, Noordwijk, The Netherlands

² TU Delft, Delft, The Netherlands

Abstract. A fast computation of the hypervolume has become a crucial component for the quality assessment and the performance of modern multi-objective evolutionary optimization algorithms. Albeit recent improvements, exact computation becomes quickly infeasible if the optimization problems scale in their number of objectives or size. To overcome this issue, we investigate the potential of using approximation instead of exact computation by benchmarking the state of the art hypervolume algorithms for different geometries, dimensionality and number of points. Our experiments outline the threshold at which exact computation starts to become infeasible, but approximation still applies, highlighting the major factors that influence its performance.

Keywords: Hypervolume indicator, performance indicators, multi-objective optimization, many-objective optimization, approximation algorithms.

1 Introduction

The *hypervolume indicator* (also known as Lebesgue measure [1] or S-metric [2]) is a popular quality measure for multi-objective optimization [3]. It was first suggested by Zitzler and Thiele [4] as the size of the objective space covered by the non-dominated solutions. It has the property of being strictly Pareto-compliant, i.e., the Pareto-optimal front guarantees the maximum possible hypervolume while any dominated set will assign a lower hypervolume [5]. Besides its application as a performance indicator, the exclusive contribution of one individual to the total hypervolume is used by Multi-Objective Optimization Algorithms (MOEA) for selection, diversification and archiving. Because well-established Pareto dominance-based MOEA like NSGA-II [6] and SPEA2 [7] deteriorate in performance as the number of objectives increases, indicator-based algorithms [8] such as SMS-EMOA [9], HypE [10] and MO-CMAES [11] provide an alternative optimization design. A fast computation of the hypervolume indicator and the contributions has thus become important for multi-objective optimization.

Bringmann and Friedrich [12] show that computing the hypervolume indicator is $\#\mathcal{P}$ -complete, which implies that there exists no polynomial-time algorithm unless $\mathcal{P} = \mathcal{NP}$. Similar hardness results hold for computing hypervolume contributions. However, the same authors also show that computing the hypervolume

is fixed parameter tractable in the average case which gives hope that exact algorithms might be useful in practice [13]. Given this opportunity, we investigate and compare the state of the art approaches for determining the least contributor: exact algorithms approaching the problem from a computational geometry standpoint and the approximation algorithm by Bringmann and Friedrich [12] which can offer a better runtime at the cost of the precision.

2 Related Work

For dimensions $d \leq 4$, hypervolume computation is in general tractable, as there exist algorithms tailored for $d = 2, 3$ and 4 [14–16]. For arbitrary dimensions, the best currently known algorithms in terms of asymptotical runtime all rely on the *Klee measure problem*, of which the computation of the hypervolume indicator is a special case. For $d \geq 7$ Bringmann [17] gives an algorithm that runs in $\mathcal{O}(n^{\frac{d+2}{3}})$ whereas Yildiz and Suri [18] are asymptotically better for $d = 4, 5, 6$ with $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$. The HOY [19] algorithm uses a space partitioning based on the divide and conquer paradigm and runs in $\mathcal{O}(n^{\frac{d}{2}} n \log n)$. The drawback of these methods are the large data structures that need to be maintained during the run. The fastest algorithm based on dimension-sweeping is the FPL algorithm by Fonseca et al. [20] that has an asymptotic complexity of $\mathcal{O}(n^{d-2} \log n)$ while using only linear space. The WFG [21] is based on using bounding boxes to determine the exclusive contributions of points, which are then used to compute the total hypervolume. Despite its asymptotic run-time of $\mathcal{O}(n^d)$, there is experimental [22] and theoretical evidence [13] that it is currently the fastest exact algorithm for higher dimensions in practice. Although a good average case complexity was proven, hypervolume computations are still tremendously time-consuming for today's computers already starting with $d = 10$. Approximation algorithms and heuristics make it possible to explore problem domains of $d \geq 10$, which often needed to be scaled down (i.e. by aggregation) before to become tractable. There exists a fully polynomial randomized approximation scheme (FPRAS) for the hypervolume indicator [23], which allows for its approximation with given precision and probability in polynomial time. Ishibuchi et al. [24] give a heuristic that uses achievement scalarizing functions to approximate the hypervolume indicator, however, no approximation ratio is known. Essential topic of our research is the algorithm of Bringmann and Friedrich [12] which combines a Monte Carlo-like sampling method with a racing approach to directly approximate the least hypervolume contributor. It is the only algorithm we are aware of that gives a guarantee that for any given $\delta, \varepsilon \geq 0$ the obtained solution is with a probability of $(1 - \delta)$ larger by at most a factor of $(1 + \varepsilon)$ than the least contributor.

Closely related to this work is the one of Bringmann et. al [25] where the empirical performance of MO-CMAES is evaluated when using the approximation algorithm as a subroutine. While it was shown that the runtime of MO-CMAES could be reduced, it is unclear which sort of geometries during evolution had to be processed by the approximation algorithm and how fast it did so.

Also, the problem with highest dimensionality chosen was $d = 8$, for which arguably an exact computation might still be feasible. As we will analyze the runtime of the approximation algorithm unbiased by any MOEA from 2 to 100 dimensions, our approach is more direct and comprehensive, extending the preliminary results from the original work. In particular, we follow the suggestion made by the authors at the end of the original work [12] to create a broader experimental setup, in which we can observe the occurrence and influence of hard cases along other factors that impact the runtime by orders of magnitude, but were never addressed in previous works before.

3 Experimental Setup

In this section we describe the datasets we used for our experiments¹. Since the hypervolume indicator computation is inherently connected with a reference point, we propose a robust selection procedure. Finally, we assess the set of the best-performing exact algorithms to provide a reliable comparison with the approximation algorithm in the next section.

3.1 Datasets

We assume minimization in our setup and that each point in the dataset is constrained to a unit box $[0, 1]^d$. We generate the datasets with three analytically-defined geometries (*Convex*, *Concave* and *Planar*). In order to show the performance of the algorithms on the maximization problems of the same geometries (and also to close the gap between the publications that assume it), we provide their inverted variants: *InvertedConvex*, *InvertedConcave* and *InvertedPlanar*.

Datasets were generated by sampling a multivariate normal distribution, similarly to how it was outlined in [12]. Let $q = (q_1, q_2, \dots, q_d)$ be a vector of d normal deviates, and $p \in \mathbb{R}^+$ a parameter of the norm in L^p space. We sample n non-dominated points using S :

$$S(q, p) = \frac{(|q_1|, |q_2|, \dots, |q_d|)}{(q_1^p + q_2^p + \dots + q_d^p)^{\frac{1}{p}}}. \quad (1)$$

For $p = 2$, $p = 1$ and $p = 0.5$ we obtain the *Convex*, *Planar* and *Concave* shapes respectively. Inverted variants of the shapes above involve the extra step of multiplication of each obtained vector by a scalar of -1 and translation by the vector $(1, 1, \dots, 1)$:

$$S_{Inverted}(q, p) = (1, 1, \dots, 1) - S(q, p). \quad (2)$$

Figure 1 visualizes the shapes for $d = 3$.

Additionally, we generate a group of *Random* datasets. *Random* dataset A (initially empty) is obtained by a repeated sampling of a point inside the box $[0, 1]^d$ and performing a test for the pairwise non-dominance with all of the previously sampled points in A . This process continues until $|A| = n$.

¹ Source code available: <https://github.com/esa/pagmo/wiki/Hypervolume>.

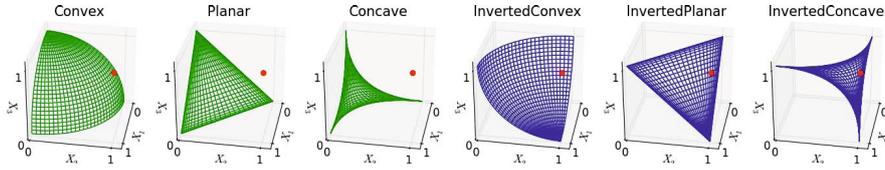


Fig. 1. Regular (green) and inverted (blue) dataset shapes, possible reference point (red) at (1, 1, 1)

Dataset size ranges from 10 to 100 points by a step of 10 and then from 100 to a 1000 points by a step of 100. Dimension ranges from 2 to 20 by a step of one, and then from 20 to 100 by a step of 10. We generate a sample of 10 datasets in each shape and for each combination of dimension and number of points. Although current state of the art in multi-objective optimization does not deal with problems of such extreme dimensionality, we are interested in presenting the empirical scaling capabilities of the algorithm itself.

3.2 Reference Point

Every hypervolume computation requires a reference point. One way of obtaining it, is simply assuming a fixed point which is guaranteed to be strongly dominated by all of the points in the set. When the hypervolume is employed for the optimization scenarios, this information might not be known upfront. In such cases, the reference point is chosen dynamically, i.e. dependent on the point set. A common approach is constructing a point out of the maxima in each coordinate (known as the *nadir point*), and offsetting it by a small constant. This assures that the points on the boundaries of the space have non-zero contribution to the total hypervolume, as it was explained in the work of Beume et al. [9]. However, a constant offset for the reference point may lead to problems, as any fixed value may be relatively large in comparison to the space boundary. In such case, border points may be influencing the hypervolume too strongly. In order to avoid that, we will shift the nadir point relatively to the boundary of the point set, as it was proposed by Knowles [26]. With N as the nadir point, and I as the ideal point (minima among all coordinates), we compute the reference point as follows:

$$R = N + \alpha \cdot (N - I). \tag{3}$$

For our experiment we assume $\alpha = 0.01$, resulting in a reference point shifted by 1% in each of the dimensions of the bounding box.

3.3 Selecting the Exact Algorithms and Experiment Design

We test selected exact hypervolume algorithms in order to determine a robust and efficient candidate, which we will use for the comparison with the approximation algorithm. We consider three dimension-specific algorithms: Dimension-sweep algorithm for $d = 2$ (HV2D), algorithm by Beume for $d = 3$ (HV3D)

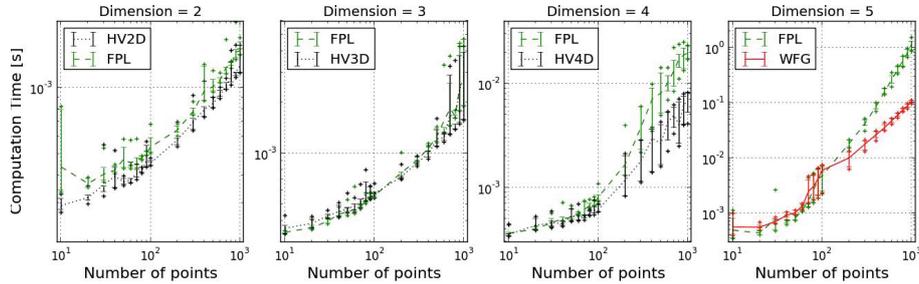


Fig. 2. Distribution of 10 hypervolume indicator computation times (*Random dataset*)

and algorithm by Guerreiro for $d = 4$ (HV4D). Besides these, we test three dimension-independent algorithms: WFG, FPL and HOY. All of the implementations of algorithms used for this research, can be found in the PaGMO library².

Figure 2 shows the median times of 10 hypervolume indicator computations for algorithms WFG, FPL, and the family of dimension-specific algorithms: HV2D, HV3D and HV4D, with whiskers describing the middle 8 runtimes (outliers are denoted with a “plus” mark). On each figure only the best and the second best performing algorithms are displayed per dimension. All of the dimension-specific algorithms tend to perform no worse in their domain than the dimension-independent ones. Out of the dimension-independent algorithms, WFG performs better than FPL and HOY on every test instance with more than 5 dimensions. For 5 dimensions, FPL performs better than WFG for 80 points or less.

Least contributor is obtained through $n + 1$ computations of the hypervolume:

$$\text{LeastContributor}(S) = \underset{p \in S}{\operatorname{argmin}}(\text{Hypervolume}(S) - \text{Hypervolume}(S \setminus \{p\})). \quad (4)$$

We improve on that by employing the algorithm by Emmerich et al. [16] for 3 dimensions, and a version of WFG optimized for the least contributor computation. Figure 3 presents the results for the least contributor computation. For $d \geq 4$ WFG outperformed all other algorithms.

We propose a group of best performing algorithms for the computation of the least contributor, which we will compare with the PaGMO implementation of the approximation algorithm by Bringmann and Friedrich [12] (to which we will refer as BFA). For that task we select all dimension-specific algorithms for $d \leq 3$ and WFG in every other case. Because BFA employs a mechanism in which difficult subproblems can be solved using the computation of the hypervolume indicator, we define a set of the best performing algorithms for the hypervolume indicator as well. For $d \leq 4$ we will use all of the dimension-specific algorithms, for $d = 5$ and $n < 80$ we will employ FPL, while the remaining cases will be handled by the WFG algorithm. We run the approximation algorithm with the parameters recommended by the authors, namely $\delta = 10^{-6}$ and $\varepsilon = 10^{-2}$. All experiments

² Available online at <https://github.com/esa/pagmo>

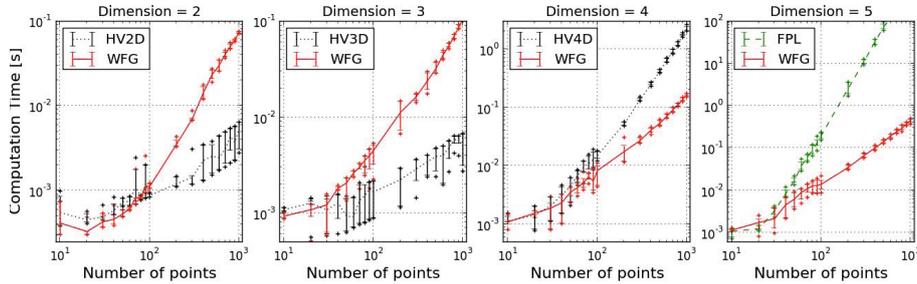


Fig. 3. Distribution of 10 least hypervolume contributor computation times (*Random* dataset)

were conducted on an Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz with 20480 KB of cache.

4 Results

This section covers the results obtained by our comparison experiment. As exact computation of the least contributor is intractable for the majority of our test cases – rendering the measurement of empirical accuracy of BFA infeasible – we have to restrict our analysis to the runtime only.

Runtime patterns derived for the exact algorithms and the BFA algorithm can be seen in Figure 4. Given the extent of the data, we terminate each computation after 30 seconds to make the experiment feasible and to show the general outlook of the runtime patterns. First, we observe that *Planar*, *Concave* and *Convex* shapes were similar in the runtime patterns they produced, thus we used *Convex* as a representative of this group of shapes. For the same reason *InvertedConvex* was chosen as the representative of the inverted variants of the shapes above. The runtime pattern of the exact algorithm was similar across all shapes, while BFA performed worse for the first group (*Planar*, *Concave* and *Convex*) when compared with their corresponding inverted variants. Figures 4(c), 4(d) and Figures 4(e), 4(f) show a significant difference in the performance of the approximation algorithm. Due to space limitation we do not provide the corresponding plots for the BFA’s performance on the *Random* shape, which was similar to the *Convex* shape. Figure 4(b) shows that using the fastest known exact algorithms can still be efficient when the dimension is no larger than 7 or when the front consists of 20 points or fewer.

To better understand the runtime pattern of BFA in Figures 4(c) and 4(d), we drop the 30 second termination criterion and rerun the experiment for 200 points until completion. Figure 5 presents the interdependence of BFA’s runtime (left plot) to the total number of Monte Carlo samplings performed by the algorithm (middle plot). We observe that for a fixed number of points, variance of the runtime increases at $d = 10$, while the average runtime slowly declines as the dimension increases (Figures 4(c) and 4(d)). Taking the exact computation usage

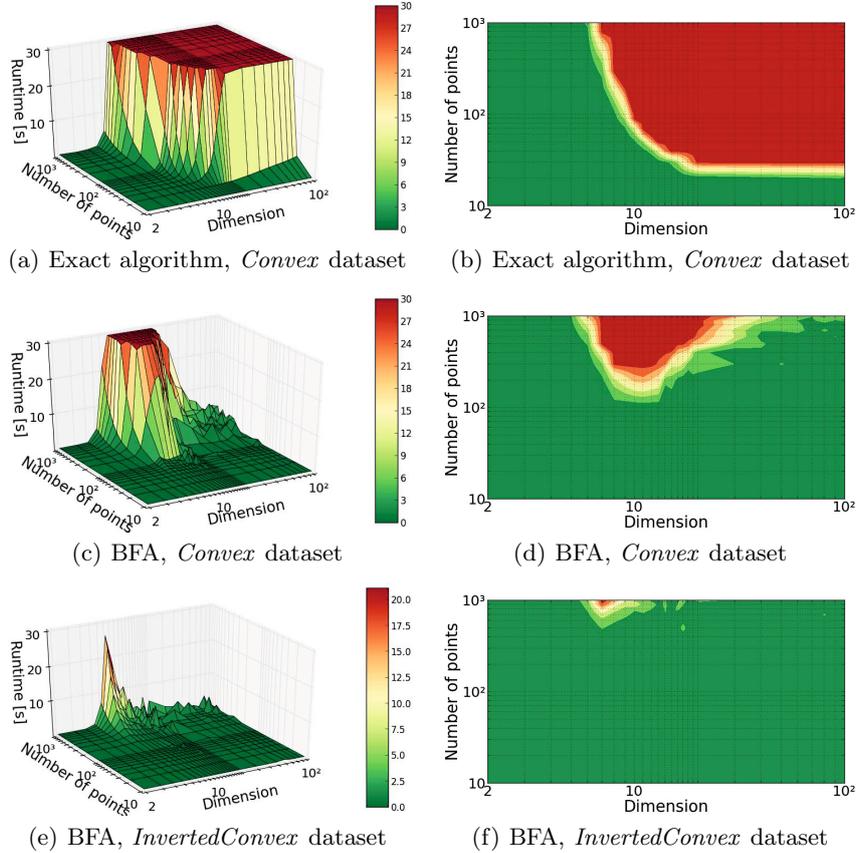


Fig. 4. Runtime of the exact algorithm – figures (a) and (b) – on *Convex* dataset, and the BFA algorithm – figures (c) to (f) – on *Convex* and *InvertedConvex* datasets

into account (rightmost plot in Figure 5) it is evident that the usage of exact computations for $d \leq 7$ amortizes the runtime in lower dimensions for BFA.

To investigate further, we pick the case with the highest runtime for each dimension and present the distribution of the number of samples over 200 points in Figure 6. We observe a dependence of the number of samples to the dimension. It is most of the time the (true) least contributor and one or two of other candidates which constitute for the majority of total number of samples, suggesting that these points remained in the race for a long time. This happens when two or more points differ very little in their contribution, which supports the impact of the *hardness of approximation*, as is was described in the original work by Bringmann and Friedrich. Surprisingly, this effect seems to be inversely proportional to the dimension (given a fixed number of points). We believe that this can be attributed to relatively sparse distribution of points as the dimension increases, leading to fewer occurrences of hard cases.

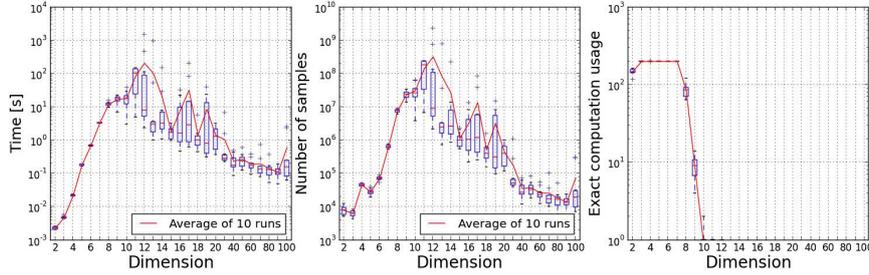


Fig. 5. Runtime of BFA algorithm (left), total number of performed sampling rounds (middle) and the number of exact computations performed by the algorithm (right) Distribution over 10 runs per dimension on *Convex* dataset of 200 points

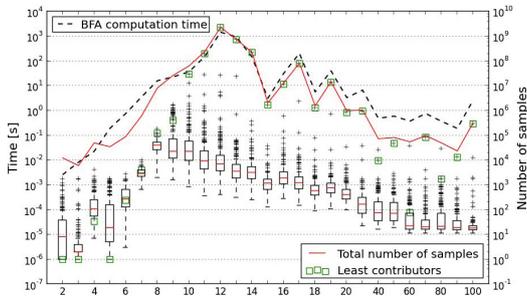


Fig. 6. Runtime of BFA algorithm (dashed line) and the distribution of the number of performed sampling rounds over 200 points (least contributors marked with a green square, red line shows the total number of samples). Maximal runtimes for each dimension picked from *Convex* datasets of 200 points.

While the high runtime has already been tied to the shape of the data, we have observed the reference point to also influence the empirical runtime performance of BFA. Figure 7 shows a runtime comparison of an exact computation using WFG and the approximation using BFA, with varying offsets applied to the nadir point. Altering the reference point influences the relative contributions of the border points, thus the least contributor can change. While the runtime of WFG seems to be indifferent to the reference point, the observed runtime of BFA spans over two orders of magnitude, in favor of smaller offsets. We suggest using a reference point relative to the size of the objective space and with a small offset (1% per objective), as we have done in our previous experiments.

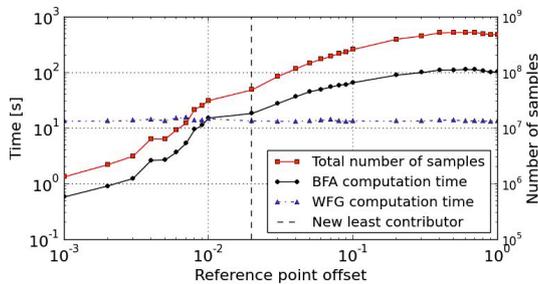


Fig. 7. Impact of the reference point offset on the runtime of WFG and BFA algorithms and the number of Monte Carlo sampling rounds. Change of the least contributor marked by a dashed line. *Concave* dataset with 100 points in 10 dimensions.

5 Conclusions

Since the currently best available exact computation algorithms quickly reach their limits for problems with 10 or more dimensions, BFA provides a superior performance as it scales much better in that regard. However, easy to overlook factors such as the geometry of the dataset or the choice of the reference point can degrade the runtime of the approximation algorithm up to two orders of magnitude, even though their observed impact on the exact methods was minimal. Although our test data was not explicitly designed to create hard cases for BFA, we observed their frequent occurrence, indicating that its runtime, while still much faster than those of exact algorithms, could be nevertheless unexpectedly high in ill-conditioned cases. By outlining the relation between the runtime of BFA and the behaviour of the underlying Monte Carlo sampling scheme, we highlight easy to overlook factors that need to be considered before employing the algorithm in practice. Taking these points into account, hypervolume approximation has a great potential for opening up previously intractable problem domains for optimization and research.

Acknowledgement. We thank Tobias Friedrich for his encouragement and helpful comments at the early stages of our research.

References

1. Fleischer, M.: The measure of Pareto optima applications to multi-objective metaheuristics. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003)
2. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63. Shaker Ithaca (1999)
3. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)
4. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
5. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
7. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm (2001)
8. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

9. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181(3), 1653–1669 (2007)
10. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
11. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation* 15(1), 1–28 (2007)
12. Bringmann, K., Friedrich, T.: Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. *Theoretical Computer Science* 425, 104–116 (2012)
13. Bringmann, K., Friedrich, T.: Parameterized average-case complexity of the hypervolume indicator. In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2013*, pp. 575–582. ACM, New York (2013)
14. Beume, N., Fonseca, C.M., López-Ibáñez, M., Paquete, L., Vahrenhold, J.: On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation* 13(5), 1075–1082 (2009)
15. Guerreiro, A.P., Fonseca, C.M., Emmerich, M.T.: A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In: *CCCG*, pp. 77–82 (2012)
16. Emmerich, M.T.M., Fonseca, C.M.: Computing hypervolume contributions in low dimensions: asymptotically optimal algorithm and complexity results. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 121–135. Springer, Heidelberg (2011)
17. Bringmann, K.: An improved algorithm for Klee’s measure problem on fat boxes. *Computational Geometry* 45(5), 225–233 (2012)
18. Yildiz, H., Suri, S.: On Klee’s measure problem for grounded boxes. In: *Proceedings of the 2012 Symposium on Computational Geometry*, pp. 111–120. ACM (2012)
19. Beume, N.: S-metric calculation by considering dominated hypervolume as Klee’s measure problem. *Evolutionary Computation* 17(4), 477–492 (2009)
20. Fonseca, C.M., Paquete, L., López-Ibáñez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 1157–1163. IEEE (2006)
21. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation* 16(1), 86–95 (2012)
22. Priester, C., Narukawa, K., Rodemann, T.: A comparison of different algorithms for the calculation of dominated hypervolumes. In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2013*, pp. 655–662. ACM, New York (2013)
23. Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry* 43(6), 601–610 (2010)
24. Ishibuchi, H., Tsukamoto, N., Sakane, Y., Nojima, Y.: Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 527–534. ACM (2010)
25. Bringmann, K., Friedrich, T., Igel, C., Voß, T.: Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence* 204, 22–29 (2013)
26. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10(1), 50–66 (2006)